# DAFTAR PUSTAKA


[1] surya.co,id, 29 Desember 2017, Curanmor di jatim menurun drastis, diakses pada tanggal 15 maret 2018, http://surabaya.tribunnews.com/2017/12/29/curanmor-di-jatim-menurun-drastis-kapolda-pertanda-masyarakat-mulai-sadar-soal-ini.

[2] Trimulyadi, 2016, Desain Dan Pembuatan Alat Pengaman Sepeda Motor Dengan Sistem Kontrol *Arduino*, Publikasi Ilmiah, Fakultas Teknik. Universitas Muhammadiyah Surakarta

[3] Sandro Lumban Tobing, 2014, Rancang Bangun Pengaman Pintu Menggunakan Sidik Jari (*Fingerprint*) Dan *Smartphone Android* Berbasis Mikrokontroler Atmega8, tanggal 13 maret 2018, *jurnal infomatika*. Universitas Tanjungpura Pontianak.

[4] Dwi Ely Kurniawan 1, Muhamad Naharus Surur2, 2016, Perancangan Sistem Pengamanan Sepeda Motor Menggunakan Mikrokontroler *Raspberry Pi* dan *Smartphone Android*, Jurnal Politeknik Caltex Riau. Politeknik Negeri Batam.

[5] Lingga Hartadi dan Dani Sasmoko, 2015, Sistem Keamanan Kendaraan Suzuki Smash Menggunakan Atmega 8 Dengan Sensor *Bluetooth Hc-6* Berbasis *Android*, *jurnal ELKOM*. Sekolah Tinggi Elektronika dan Komputer.

[6] Adhitya Wishnu Wardhana dan Yudi Prayudi, 2008, Penggunaan Metode *Templete Matching* Untuk Identifikasi Kecacatan Pada PCB, Jurnal UII, Fakultas Teknologi Industri. Universitas Islam Indonesia.

[7] Eni Yuliza dan Toibah Umi Kalsum, Februari 2015, Alat Keamanan Pintu Brankas Berbasis Sensor Sidik Jari dan *Passoword* Digital

Dengan Menggunakan Mikrokontroler Atmega16, Jurnal Media Infotama. Universitas Dehasen Bengkulu.

[8] Juli Dian Purbani, 2010, Pembuatan Mesin Identifikasi Sidik Jari Sebagai Kunci Pengaman Pintu, Ilmu Komputer. Universitas Sebelas Maret Surakarta.

[9] Idris Setiawan, 2015, Sistem Pengaman Pintu Rumah Menggunakan Sensor Sidik Jari (*Fingerprint*), Fakultas Teknik. Universitas Negeri Semarang,Tugas Akhir

[10] Alan Burhannudin, 2017, Optimalisasi Identifikasi Sidik Jari Menggunakan Metode *Neural Network* Pada Sistem Keamanan Sepeda Motor, Teknik Mekatronika. Politeknik Negeri Batam, Tugas Akhir

[11] Bowo Leksono, Achmad Hidayatno dan R. Rizal Isnanto, 2011, Aplikasi Metode *Template Matching* untuk Klasifikasi Sidik Jari, Jurnal Undip. Fakultas Teknik Universitas Diponogoro Semarang.

# 1. LAMPIRAN 1

**Progam simulasi matlab**

```
function varargout = match(varargin)
% MATCH M-file for match.fig
%      MATCH, by itself, creates a new MATCH or raises the existing
%      singleton*.
%      H = MATCH returns the handle to a new MATCH or the handle to
%      the existing singleton*.
%      MATCH('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MATCH.M with the given input
arguments.
%      MATCH('Property','Value',...) creates a new MATCH or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before match_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to match_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help match
% Last Modified by GUIDE v2.5 03-Aug-2014 21:13:44
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @match_OpeningFcn, ...
                   'gui_OutputFcn',  @match_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before match is made visible.
function match_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to match (see VARARGIN)
 % Choose default command line output for match
handles.output = hObject;
 % Update handles structure
guidata(hObject, handles);
```

```
 % UIWAIT makes match wait for user response (see UIRESUME)
% uiwait(handles.figure1);

 % --- Outputs from this function are returned to the command line.
function varargout = match_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 [name_file1,name_path1] = uigetfile( ...
    {'*.bmp;*.jpg;*.tif','Files of type (*.bmp,*.jpg,*.tif)';
    '*.bmp','File Bitmap (*.bmp)';...
    '*.jpg','File jpeg (*.jpg)';
    '*.tif','File Tif (*.tif)';
    '*.*','semua file (*.*)'},...
    'Buka Citra asli');
 if ~isequal(name_file1,0)
    handles.im1 = imread(fullfile(name_path1,name_file1));
    guidata(hObject,handles);
    axes(handles.axes1);
```

```
    imshow(handles.im1);title('database');
else
    return;
end
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[name_file1,name_path1] = uigetfile( ...
    {'*.bmp;*.jpg;*.tif','Files of type (*.bmp,*.jpg,*.tif)';
    '*.bmp','File Bitmap (*.bmp)';...
    '*.jpg','File jpeg (*.jpg)';
    '*.tif','File Tif (*.tif)';
    '*.*','semua file (*.*)'},...
    'Buka Citra asli');
 if ~isequal(name_file1,0)
    handles.im2 = imread(fullfile(name_path1,name_file1));
    guidata(hObject,handles);
    axes(handles.axes3);
    imshow(handles.im2);title('finger');
else
    return;
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
p=handles.im1;
p1=im2bw(p);
p2=bwarea(p1);
set(handles.edit2,'String',p2);
 q=handles.im2;
q1=im2bw(q);
q2=bwarea(q1);
set(handles.edit3,'String',q2);

p1=im2bw(p);
q1=im2bw(q);
m=p1-q1;
axes(handles.axes2);
imshow(m);title('matching template');

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 p=handles.im1;
p1=im2bw(p);
p2=bwarea(p1);
set(handles.edit2,'String',p2);
 q=handles.im2;
q1=im2bw(q);
q2=bwarea(q1);
```

```
set(handles.edit3,'String',q2);

p1=im2bw(p);
q1=im2bw(q);
m=p1-q1;
axes(handles.axes2);
imshow(m);title('matching template');

p=handles.im1;
p1=im2bw(p);
p2=bwarea(p1);
q=handles.im2;
q1=im2bw(q);
q2=bwarea(q1);
x=p2;
m=(q2+p2)/2;
persen=(m/x)*100;
set(handles.edit1,'String',persen);
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
 % --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
 % --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


 % --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 close
```

# LAMPIRAN 2

**Program arduino IDE**

**1. Enroll(pendaftaran sidik jari)**

```
#include <Adafruit_Fingerprint.h>
// On Leonardo/Micro or others with hardware serial, use those! #0 is green wire, #1 is white
// uncomment this line:
// #define mySerial Serial1
// For UNO and others without hardware serial, we must use software serial...
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
// comment these two lines if using hardware serial
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
uint8_t id;
void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...

  delay(100);
  Serial.println("\n\nAdafruit Fingerprint sensor enrollment");
  // set the data rate for the sensor serial port
  finger.begin(57600);
    if (finger.verifyPassword()) {
```

```
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }
}
uint8_t readnumber(void) {
  uint8_t num = 0;
    while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}
void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println("Please type in the ID # (from 1 to 127) you want to save
this finger as...");
  id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);
  while (!  getFingerprintEnroll() );
}
uint8_t getFingerprintEnroll() {
```

```
int p = -1;
Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.println(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}
// OK success!
p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
```

```
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}
  Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
```

```
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}
// OK success!
p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
```

```
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}
  // OK converted!
Serial.print("Creating model for #");  Serial.println(id);
  p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}
  Serial.print("ID "); Serial.println(id);
p = finger.storeModel(id);
```

```
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}
```

**2. Fingerprint(pemberian perintah pada arduino)**

```
#include <Adafruit_Fingerprint.h>
// On Leonardo/Micro or others with hardware serial, use those! #0 is green wire, #1 is white
// uncomment this line:
// #define mySerial Serial1
// For UNO and others without hardware serial, we must use software serial...
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
// comment these two lines if using hardware serial
```

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2,3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
int relay = 4;
void setup()
{
  pinMode(relay,OUTPUT);
  pinMode(6,OUTPUT); //lampu
  pinMode(7,OUTPUT); //buzzer
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");
  // set the data rate for the sensor serial port
  finger.begin(57600);
     if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }
  finger.getTemplateCount();
  Serial.print("Sensor contains "); Serial.print(finger.templateCount);
Serial.println(" templates");
  Serial.println("Waiting for valid finger...");
}
void loop()                     // run over and over again
{digitalWrite (7,LOW);
```

```
 getFingerprintIDez();
 delay(50);            //don't ned to run this at full speed.
}
uint8_t getFingerprintID() {
 uint8_t p = finger.getImage();
 switch (p) {
  case FINGERPRINT_OK:
   Serial.println("Image taken");
   break;
  case FINGERPRINT_NOFINGER:
   Serial.println("No finger detected");
   return p;
  case FINGERPRINT_PACKETRECIEVEERR:
   Serial.println("Communication error");
   return p;
  case FINGERPRINT_IMAGEFAIL:
   Serial.println("Imaging error");
   return p;
  default:
   Serial.println("Unknown error");
   return p;
 }
 // OK success!
 p = finger.image2Tz();
 switch (p) {
  case FINGERPRINT_OK:
   Serial.println("Image converted");
   break;
```

```
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}
  // OK converted!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
  Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Did not find a match");
  return p;
} else {
  Serial.println("Unknown error");
```

```
   return p;
  }
     // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
    p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;
  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;
      // found a match!
  digitalWrite(4,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(4,HIGH);
  delay(1000);
  digitalWrite(7,LOW);
  delay(100000000);
  Serial.print("rovek Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
```